Guidelines for Integrating DiskOnChip in a Host System

Application Note, August 2004
Arie Tal (arie.tal@m-systems.com)

1. SCOPE

This document provides a quick overview of DiskOnChip technology and a fundamental review of the tasks required to implement DiskOnChip on-board.

This document does not take the place of reading the relevant DiskOnChip data sheet. It does, however, offer a basic understanding of DiskOnChip and the major steps required when integrating DiskOnChip in a host system.

2. DISKONCHIP

DiskOnChip is a fast, high-capacity, cost-effective, reliable, nonvolatile memory solution. It is based on an ideal mix of hardware, software and best-of-breed flash technology, such as Multi-Level Cell (MLC) NAND. This unique combination allows engineers to take advantage of the cost benefits of MLC NAND technology while maintaining extremely high reliability and boosting read and write performance.

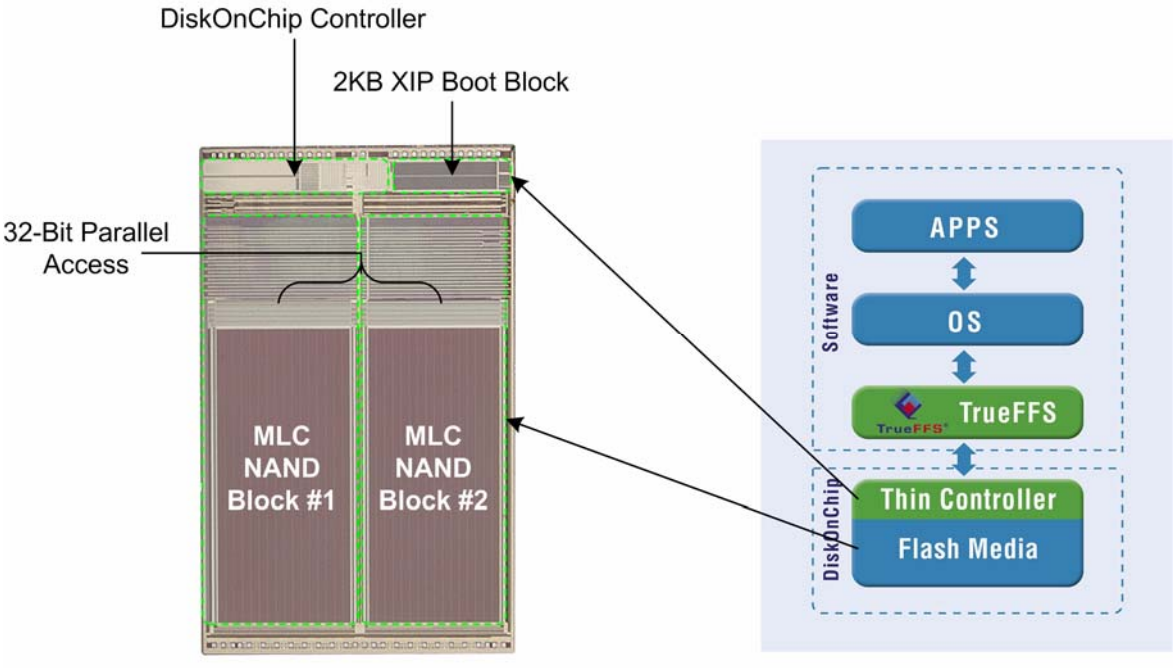

Figure 1: DiskOnChip Silicon Die and Block Diagram

2.1 NAND Flash Memory

DiskOnChip is based on the best flash technology available to date. It has featured NAND flash for embedded storage since 1996. Today DiskOnChip is the only solution that implements MLC NAND for local (embedded) code and data storage. In addition, the media geometry inside DiskOnChip enables two MLC NAND flash blocks to be accessed in parallel, providing a substantial improvement in performance over the capability of the basic technology.

2.2 Thin Controller

To minimize the cost of the Bill of Materials (BOM), the DiskOnChip controller is located on the same silicon die as the flash media, yet only comprises up to 5% of the die. All access to the NAND flash media and to DiskOnChip is handled through this embedded controller. This design offers the following improvements over basic NAND technology:

- NOR/SRAM interface, compatible with most CPUs on the market
- 2KB eXecute In Place (XIP) Boot Block to enable NOR-less systems
- Built-in Error Detection Code (EDC) hardware for MLC NAND flash
- Power management that reduces consumption in Deep Power-Down mode to only 10 µA
- Support for cascaded devices: Up to four DiskOnChip devices can be connected on the same 8KB memory window
- Performance improvement features: Hardware DMA for read (up to 64KB), burst read (up to 1KB), interrupt controller, Turbo mode, concurrent NAND plane access, and automatic page increment
- Advanced security features: Unique ID (UID), 6KB One-Time Programmable (OTP) area, two configurable read write protected partitions, etc.
- Byte swapper for Big Endian platforms

2.3 TrueFFS for DiskOnChip

TrueFFS is highly advanced, patented flash management software that serves as the DiskOnChip driver and provides a software interface between the host operating system (OS) and DiskOnChip. It interfaces with the host file system and presents DiskOnChip as a standard hard drive, making the flash management details transparent. TrueFFS can even replace the host file system with its own rugged, Microsoft-FAT compatible file system solution, called SureFS.

TrueFFS has become the standard software for PCMCIA memory cards since 1995. TrueFFS has been compatible with Microsoft OSs since 1997, and has been offered for licensing by WindRiver since 1999. TrueFFS is integrated in major mobile OSs such as Palm, Symbian, and Windows CE/PocketPC 200x, and supports many others such as Nucleus, OSE, QNX, DOS, and ThreadX.

2.4 Electrical Interface

DiskOnChip provides the benefit of NAND flash via a convenient and legacy-compatible NOR/SRAM interface (see Figure 2), which allows for easy electrical integration with almost every processor on the market. This includes high-end processors such as TI OMAP and Intel XScale, as well as low-end processors such as Agere Vienna, Infineon E-Gold, and Analog Devices AD6525. A DiskOnChip symbol file and IBIS models (OrCAD-compatible) are available in M-Systems' website developer's area (www.m-systems.com/developer).

Figure 2: Electrical Interface Comparison

2.5 Memory Mapping

DiskOnChip occupies an 8KB memory window, which is mapped in its entirety to the DiskOnChip controller. All access to the flash media on DiskOnChip is therefore handled via the controller. The 8KB window is mapped as shown in Figure 3.

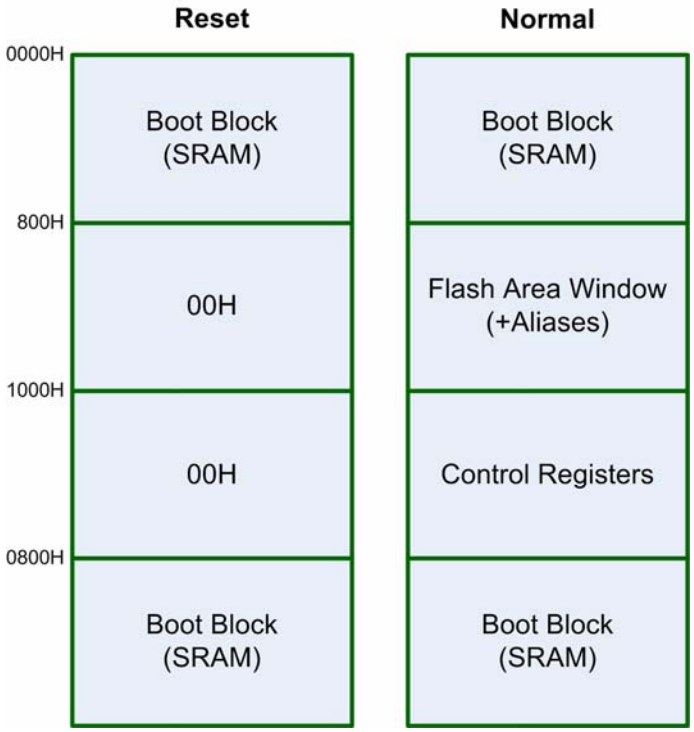

Figure 3: DiskOnChip 8KB Memory Window

The 2KB located in the lower and upper ends of the DiskOnChip memory window are mapped to the internal SRAM Boot Block. These 2KBs have XIP functionality, and can be used to run a small boot program. The 4KB between 0x800 and 0x1800 reflect various registers in the DiskOnChip controller. The flash media is accessed by applying a command to the control registers and then reading/writing data through the flash access registers in the controller.

2.6 Accessing DiskOnChip

The following process describes a typical write command to DiskOnChip:

1. Write the two physical page addresses (interleave 2) to the address register.
2. Write a flash clear command to clear the page buffer.
3. Initialize the EDC machine.
4. Write user data in 256-word chunks.
5. Write 7-byte chunks of control data
6. Read the Hamming and BCH bytes from the EDC machine and write them.
7. Flush the page buffer to the flash array.
8. Poll the busy/ready register.

This means that the minimum data transfer size (read and write) from and to DiskOnChip is one sector (512 bytes). Even if the relevant data occupies only one byte, an entire page is written to the flash media.

The exact register settings and commands required to access DiskOnChip are not relevant to developers, as DiskOnChip access is managed by the TrueFFS driver provided free to DiskOnChip customers.

2.6.1 Helpful Tips

- Although DiskOnChip features a NOR-like interface, only the Boot Block has XIP functionality.
- The DiskOnChip memory window occupies only 8KB.
- Up to four DiskOnChip devices may be cascaded in the same memory window.
- DiskOnChip should NOT be accessed using small buffers.
- The ideal buffer size is between 512 bytes and 2KB.

2.7 Booting from DiskOnChip

2.7.1 Overview

Since NAND flash technology does not support XIP functionality, code cannot be directly executed from DiskOnChip. To overcome this, DiskOnChip G3/P3 includes 2KB of SRAM built into its controller. When a RESET signal is negated to DiskOnChip, the DiskOnChip controller fetches the 2KB Initial Program Loader (IPL) from the flash media and loads it to the SRAM boot block. While this is happening, the DiskOnChip BUSY signal is low. Once the IPL is loaded, the busy signal goes high, indicating to the CPU to start reading the code from the DiskOnChip boot code.

The 2KB IPL is too small to include the entire boot code. As such, the IPL initializes the system RAM, and then copies the Secondary Program Loader (SPL), which is the reminder of the boot code, from DiskOnChip to the system RAM. The SPL then completes the system initialization process and copies (shadows) the OS image to the RAM and executes it.

2.7.2 Boot Process

The following process describes booting from DiskOnChip:

1. Upon power-up, the IPL is loaded to the DiskOnChip RAM.
2. The DiskOnChip BUSY signal goes high.
3. The CPU executes the IPL, which copies the SPL to the system RAM.
4. The SPL completes the system initialization process.
5. The SPL reads the OS image from DiskOnChip to the system RAM using TrueFFS.
6. The OS image runs from the system RAM.

2.7.3 Additional Information

- The DiskOnChip 2KB SRAM timing is 90 nsec (vs. the DiskOnChip controller access timing of 55 nsec).
- Connecting the DiskOnChip BUSY signal to the CPU RESET pin keeps the CPU in reset while DiskOnChip is loading the IPL to the SRAM. Alternatively, you can make sure the CPU does not access the DiskOnChip IPL before the maximum loading time has expired, as defined in the DiskOnChip data sheet (about 1 msec).
- The DiskOnChip IPL code is not capable of fixing bit flips. The SPL, however, does provide this capability. The IPL must never be used to load the entire OS image, but rather should be used only to copy the SPL to the system RAM.
- Shadowing the OS to the system RAM using DMA dramatically reduces the overall boot time.

3. INTEGRATING DISKONCHIP IN A HOST SYSTEM

When working with DiskOnChip, the integration process in a host system is simple. First, design DiskOnChip into the host board based on the specifications provided in the DiskOnChip data sheet. Then, integrate the TrueFFS driver into the OS image. If DiskOnChip is used as a boot device (i.e. in a NOR-less system), the boot loader may require some modification.

This section reviews the requirements for accomplishing the DiskOnChip integration process, and describes the issues that are important for designers and require careful attention. Common mistakes are also discussed, and tips are provided to ensure rapid and successful integration.

3.1 Required Tools

The following tools are required to successfully integrate DiskOnChip in a platform:

- Hardware schematic
- TrueFFS low-level formatting tool
- TrueFFS driver/file system for your OS
- TrueFFS-based tool for reading a master image from DiskOnChip

If DiskOnChip is used as a boot device, you will also need:

- IPL example for your processor
- TrueFFS SDK or BDK for writing your SPL

In addition, if DiskOnChip has to be updated after it has been soldered on-board, you must either port TrueFFS tools for your JTAG or USB solutions, or use a JTAG/USB tool from one of the various companies that already support DiskOnChip.

3.2 Verifying the Hardware Design

3.2.1 General Information

Connecting DiskOnChip to most CPUs is relatively simple, and conforms to the same rules for connecting any other SRAM or NOR device. DiskOnChip has some additional pins/balls that provide added functionality, such as DMARQ#, IRQ#, LOCK, ID, and BUSY#, which are explained in detail in the DiskOnChip data sheets.

To simplify the design effort and avoid mistakes, a ready-made DiskOnChip G3/P3 OrCAD symbol is available for download from the M-Systems' website (www.m-systems.com) for importing into your hardware design software. You can even download a ready-made schematic that has all recommended static connections (resistors, capacitors) already designed.

To verify your hardware design before your full software environment is ready, you can manually access the registers in the DiskOnChip controller. The DiskOnChip data sheet includes a test sequence for reading the DiskOnChip Chip ID using direct register access. Application note AP-DOC-0204, *Testing DiskOnChip G3/P3*, also provides instructions for testing DiskOnChip in the hardware design, and explains how to read from and write to the DiskOnChip SRAM to verify that all signals are properly connected.

A basic test code in C is available from your local M-Systems Field Application Engineer (FAE).

3.2.2 DiskOnChip in Big Endian Designs

DiskOnChip is a Little Endian device. It should **not** to be regarded as a standard memory device that is indifferent to Endian issues. Managing Big/Little Endian compatibility is confusing, and beyond the scope of this application note. It is therefore critical to review application note AP-DOC-0504, *Big/Little Endian Byte Order*, before implementing your design. It is also strongly recommended to send your completed hardware schematics to your M-Systems FAE for review.

3.2.3 Final Design Check

Before progressing to the next stage of the integration process, review these questions:

- Did you connect the pull-up resistors and decoupling capacitors as described in the relevant DiskOnChip data sheet?
- Are the ID pins connected properly?
- If you are using a Big Endian CPU in your design, have you read and implemented the instructions in application note AP-DOC-0504, *Big/Little Endian Byte Order*?
- Have you sent your schematic (or the DiskOnChip section of the schematic) to your M-Systems FAE for review?

When DiskOnChip is connected as a boot device, also use these questions as a guideline before proceeding:

- When using 1.8V IO, did you verify that the reset signal to DiskOnChip is not the result of an RC circuit, and is shorter than 15 nsec, in any case?
- Is the CPU configured to use DiskOnChip as the default boot device?
- Did you make sure that the CPU does not access DiskOnChip before the IPL has been loaded to SRAM (~1 msec), or alternatively, did you connect the DiskOnChip BUSY signal to the CPU RESET IN signal?
- Did you verify that the bus is connected properly (8-bit/16-bit/32-bit mode)?
- Did you verify that the access time during the boot stage is appropriate for accessing the SRAM block?

3.3 Formatting DiskOnChip

3.3.1 Overview

After the hardware design is complete and has been verified, the first step in preparing DiskOnChip for your design environment is formatting. The TrueFFS software driver that manages DiskOnChip expects to find certain essential data structures on DiskOnChip that define the physical partitions and where to place management information on-board.

The M-Systems DFORMAT utility (available in Windows and DOS versions) is required to format DiskOnChip. The formatting can be done on any of the following:

- A PC, using a PCI or ISA evaluation board (EVB) from M-Systems.

- On-board your platform using a JTAG/USB solution, ported to work with M-Systems' XP utilities (the XP tools run on the host PC and communicate with DiskOnChip via the JTAG solution).
- On-board and running in your actual platform memory, and using the platform's processor. The tool can be compiled and downloaded to your platform using your local IDE, such as Code Composer, Code Warrior, or other solutions.

To accelerate and simplify development, it is recommended to obtain a simplified version of the M-Systems formatter in source code from your M-Systems FAE.

3.3.2 Formatting Options

TrueFFS supports many features, many of which must be initialized and prepared during the formatting stage. Some examples are:

- **Partitioning**: DiskOnChip can contain up to four partitions, of which three may be binary partitions (for OS image, boot loader, etc.) and at least one MUST be a disk partition (emulating a block device and providing sector-level access).
- **Protection**: You may protect up to two partitions to be read only or write only.
- **Boot**: The formatter should be used to place the IPL code on the DiskOnChip media. This operation can be done separately.
- **Partial update**: DFORMAT allows updating the content of the binary partition(s) without erasing other partitions or their contents.
- **Special features**: DFORMAT supports many special features that enable various environmental requirements, such as PC-specific options, Microsoft-specific requirements, etc.

Please refer to the DFORMAT user manual (*DiskOnChip Software Utilities for TrueFFS*) for the full list of formatting options.

3.4 Installing the TrueFFS Driver

3.4.1 Standard Systems

After formatting DiskOnChip, the TrueFFS driver must be installed in the OS image. This driver usually resides below the file system driver, and provides block device emulation (sector-level read/write functionality, just like a hard drive). TrueFFS implementations are already available for many OSs, such as Windows CE, Linux, Nucleus, DOS, PALM, Symbian, VxWorks, QNX, and others. In each OS implementation, TrueFFS interfaces with the local file system. If your OS is not supported, it is recommended to contact your M-Systems FAE so support can be added. This process is time-consuming (a license agreement must be signed with the OS vendor, the environment must be studied, TrueFFS must be ported, and the testing/qualification process must be completed prior to release), so it is important to initiate it as early as possible.

Once a driver is available, the installation process is simple. Instructions are provided with an enclosed manual and/or readme.txt file, and include customization options so TrueFFS can be customized to suit your particular platform needs.

3.4.2 Proprietary Systems

When using a proprietary OS, you may port TrueFFS, using the TrueFFS SDK, to interact with your file system, using the source code of one of our existing drivers as an example. In any case, it is recommended to consult with an M-Systems FAE prior to starting such a project.

3.4.3 SureFS File System

TrueFFS includes a patented, Microsoft-FAT compatible, rugged file system called SureFS. SureFS insures file-system integrity in cases of sudden power failure. This means that when SureFS is used, your system will always boot to a steady state, and files stored on DiskOnChip will always be accessible (except for data written during the power failure itself). SureFS is self-repairing, and does not require utilities such as ScanDisk to fix corrupted data.

SureFS is available as an installable module for some OSs (for example, Windows CE and Symbian), replacing the standard file system offered by the OS vendor.

SureFS can also be used to replace the file system used in proprietary systems, by simply wrapping its API with those of the original file system. If your system has no file system at all, use the SureFS API alone. In fact, this is the preferred integration solution by most projects using proprietary OSs or file systems.

SureFS can also run alongside other file systems that may be in use for other devices in the system or other partitions on DiskOnChip.

Notes: 1. Read the *TrueFFS SDK* quick reference guide for a brief but comprehensive overview of TrueFFS capabilities, features, API, and customization options.

2. If you must customize TrueFFS to suit your platform, apply modifications only in the **`FLSYSTEM.C`** or **`.H`** and the **`FLCUSTOM.C`** or **`.H`** files. Do not modify the TrueFFS core files without first consulting with an M-Systems FAE.

3.5 Writing a Boot Loader for DiskOnChip

As explained in Section 2.7, booting from DiskOnChip requires splitting the standard boot code into two parts, the Initial Program Loader (IPL) and the Secondary Program Loader (SPL). The IPL resides in the Programmable Boot Block, and launches the SPL that resides on DiskOnChip.

3.5.1 Initial Program Loader (IPL)

The IPL initializes DiskOnChip and the system RAM, and then copies the SPL to the SDRAM to complete the boot process. A sample IPL is available from your M-Systems' FAE. The IPL should be customized to suit your specific platform.

The IPL is too small to include ECC code. When copying the SPL to the system RAM, the EDC machine on DiskOnChip detects any bit flips in the NAND media. If a bit flip is detected, the IPL jumps to a second copy of the SPL and copies the required page. The IPL then returns to the next page in the original SPL and continues copying to RAM. This dramatically increases the reliability of the boot process (uncorrectable bit flip probability drops from 10^{-9} for standard NAND to 10^{-18}).

The SPL includes ECC code, and can read the entire OS image reliably.

3.5.2 Secondary Program Loader (SPL)

The SPL completes the platform initialization process and then copies (shadows) the OS image to RAM. The system initialization code is the same as the code used in a NOR-based system (except for the RAM initialization performed during the IPL stage). To copy the OS image to RAM, however, the NAND media on DiskOnChip must be accessed, so M-Systems code must be included in your SPL. This code is developed based on the full TrueFFS SDK package if your OS resides on a disk partition on DiskOnChip, or the compact BDK package if your OS resides on a binary partition on DiskOnChip.

Before placing the boot loader on DiskOnChip using the DFORMAT utility, you must first create an expanded SPL file that includes two copies, as described in Section 3.5.1, using the DEXPAND utility that is available from your M-Systems FAE.

3.5.3 Tips for Writing the Boot Loader

It is helpful to follow these guidelines when writing the boot loader:

- When possible, DiskOnChip should be placed on-board in a ZIF socket. This enables replacing a device, reformatting it, and extracting/placing an image off-board without expensive and complicated JTAG or USB tools.
- The IPL must **always** load an SPL. Only the SPL should load the OS image, as only the SPL can correct bit flips.
- The SPL must be expanded using DEXPAND to eliminate boot error probability.
- If your processor jumps to DiskOnChip after deep sleep as well as after boot (as with XScale products), write your IPL accordingly.

4. IMPROVING PERFORMANCE

After the basic integration process is complete, read and write performance is generally tested. Since performance is platform-dependent, your results may not match (for better or worse) the figures quoted in the DiskOnChip data sheet. To optimize DiskOnChip on your platform for best performance, the following factors should be taken into consideration:

- CPU computing power
- Traffic controller optimization
- TrueFFS configuration
- Bus timing (access time) optimization
- Efficiency of operating system/file system
- Whether or not other applications are running simultaneously
- Testing environment and tools

For example, if DiskOnChip is receiving a back-to-back cycle time of 1 µsec, its maximum theoretical performance is limited to 2 MB/sec. After taking into account software and device overheads, the practical performance may be reduced to ~1.2-1.5 MB/sec for read operations. Improving the access time in this case has almost no effect, as it is negligible compared to the cycle time.

To gain a better understanding of this subject and to learn how to optimize DiskOnChip performance, see application note AP-DOC-0704, *Improving DiskOnChip Performance*.

5. PROGRAMMING/MASS PRODUCTION

After the development stage is complete, your project is ready to move to the mass production stage. You should not wait for the end of the development stage to begin preparations for the mass-production stage, but should begin planning during the initial phase of the development stage.

You must obtain tools for the following DiskOnChip mass production requirements:

1. Reading a master image from a master DiskOnChip device
2. Programming the master image to virgin DiskOnChip devices on the final platform
3. Verifying programming
4. Partial updating of the image after general programming is complete

If your project implements DiskOnChip only as a hard drive, then the above requirements may not apply. It is likely that DiskOnChip will only require basic formatting and at most, some additional files to be copied using USB sync software or similar.

5.1 Reading the Master Image

M-Systems provides the DIMAGE utility, which reads the contents of a master DiskOnChip and creates a virtual image that excludes the physical bad blocks. This image is then used to program virgin DiskOnChip devices. DIMAGE is available in DOS and Windows (DLL) versions.

If you can place the master DiskOnChip device on a PC platform (using an evaluation board), you can run DIMAGE in DOS or Windows to extract the master image. If you cannot remove DiskOnChip from the target platform, use the DLL version with your JTAG or USB tool. DIMAGE is already supported by several JTAG and USB debug tools, and can be ported to other solutions relatively easily.

5.2 Programming the Master Image

In addition to reading the master image from DiskOnChip, DIMAGE can place this image on a target DiskOnChip device. However, DIMAGE is seldom used “as is” in mass production. In most cases, a programming machine is used. To see a list of programming machine vendors that support DiskOnChip, go to http://www.m-systems.com/content/Developer/MassProd.asp. It is strongly recommended to check with your M-Systems FAE regarding other available solutions when you are ready to move to mass production.

To program the master image to DiskOnChip after it is soldered on-board your platform, DIMAGE must be ported to work in conjunction with a JTAG or USB tool. A list of JTAG/USB solutions that already support our tools is available at the link provided above.

5.3 Verifying the Programming

There are many ways to verify that an image was properly written. However, the most common method is to simply run some sort of checksum and compare it to the expected checksum that was calculated earlier.

5.4 Partial Image Update

In many cases, the image on DiskOnChip will require modifications after the mass production stage. This may be due to software updates, language file uploads, customer-specific content uploads and bug fixes. The complexity of this partial update varies from extremely simple to extremely complicated, depending on the image structure and content update required.

Most such updates can be done using the DIMAGE and DFORMAT tools. However, it is recommended to describe your particular situation to an M-Systems FAE, who can recommend the most efficient method for performing the partial update.

6. ADDITIONAL DOCUMENTATION

It is recommended to further research the various subjects discussed in this application note. Most relevant documentation is located on the developer's pages of the M-Systems website (http://www.m-systems.com/developer), such as:

- DiskOnChip data sheets
- Integrating DiskOnChip with various platforms (such as XScale, Emblaze, MX2.1, Big Endian, and OMAP)
- Improving performance
- Specific DiskOnChip features (such as Deep Power-Down mode, protection, boot, and DMA)
- Programming

Most software-related manuals are enclosed with the relevant software package. For example:

- A manual explaining integration with Windows CE, including BinFS issues, is enclosed with the TrueFFS driver for Windows CE (some documents apply to the source code version only).
- A full TrueFFS manual is enclosed with the TrueFFS SDK developer guide (the quick reference guide can be downloaded directly from our website).
- User manuals for TrueFFS drivers for Nucleus/VxWorks/QNX/etc. are enclosed with the driver package.

It is also recommended to check with your M-Systems FAE for additional information that is not on the website (such as sample source code and unofficial tools that may be relevant for your project).

HOW TO CONTACT US

USA

M-Systems Inc.
8371 Central Ave, Suite A
Newark CA 94560
Phone: +1-510-494-2090
Fax: +1-510-494-5545

China

M-Systems China Ltd.
Room 121-122
Bldg. 2, International Commerce & Exhibition Ctr.
Hong Hua Rd.
Futian Free Trade Zone
Shenzhen, China
Phone: +86-755-8348-5218
Fax: +86-755-8348-5418

Japan

M-Systems Japan Inc.
Asahi Seimei Gotanda Bldg., 3F
5-25-16 Higashi-Gotanda
Shinagawa-ku Tokyo, 141-0022
Phone: +81-3-5423-8101
Fax: +81-3-5423-8102

Europe

M-Systems Ltd.
7 Atir Yeda St.
Kfar Saba 44425, Israel
Tel: +972-9-764-5000
Fax: +972-3-548-8666

Taiwan

M-Systems Asia Ltd.
14 F, No. 6, Sec. 3
Minquan East Road
Taipei, Taiwan, 104
Tel: +886-2-2515-2522
Fax: +886-2-2515-2295

Internet

www.m-systems.com

General Information

info@m-sys.com

Sales and Technical Information

techsupport@m-sys.com

This document is for information use only and is subject to change without prior notice. M-Systems Flash Disk Pioneers Ltd. assumes no responsibility for any errors that may appear in this document. No part of this document may be reproduced, transmitted, transcribed, stored in a retrievable manner or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without prior written consent of M-Systems.

M-Systems products are not warranted to operate without failure. Accordingly, in any use of the Product in life support systems or other applications where failure could cause injury or loss of life, the Product should only be incorporated in systems designed with appropriate and sufficient redundancy or backup features.

Contact your local M-Systems sales office or distributor, or visit our website at www.m-systems.com to obtain the latest specifications before placing your order.

© 2004 M-Systems Flash Disk Pioneers Ltd. All rights reserved.

M-Systems, DiskOnChip, DiskOnChip Millennium, DiskOnKey, DiskOnKey MyKey, FFD, Fly-By, iDiskOnChip, iDOC, mDiskOnChip, mDOC, Mobile DiskOnChip, Smart DiskOnKey, SmartCaps, SuperMAP, TrueFFS, uDiskOnChip, uDOC, and Xkey are trademarks or registered trademarks of M-Systems Flash Disk Pioneers, Ltd. Other product names or service marks mentioned herein may be trademarks or registered trademarks of their respective owners and are hereby acknowledged. All specifications are subject to change without prior notice.